https://doi.org/10.1016/j.aei.2020.101037

A smart surface inspection system using faster R-CNN in cloud-edge computing environment

Yuanbin Wang[a,b,*], Minggao Liu[b], Pai Zheng[c], Huayong Yang[a], Jun Zou[a,*]

[a] *State Key Laboratory of Fluid Power and Mechatronic Systems, Zhejiang University, Hangzhou 310027, China*

[b] *Wuxi Xuelang Digital Manufacturing Technology LLC, Wuxi 214000, China*

[c] *Department if Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, HKSAR, China*

Abstract

Automated surface inspection has become a hot topic with the rapid development of machine vision technologies. Traditional machine vision methods need experts to carefully craft image features for defect detection. This limits their applications to wider areas. The emerging convolutional neural networks (CNN) can automatically extract features and yield good results in many cases. However, the CNN-based image classification methods are more suitable for flat surface texture inspection. It is difficult to accurately locate small defects in geometrically complex products. Furthermore, the computational power required in CNN algorithms is usually high and it is not efficient to be implemented on embedded hardware. To solve these problems, a smart surface inspection system is proposed using faster R-CNN algorithm in the cloud-edge computing environment. The faster R-CNN as a CNN-based object detection method can efficiently identify defects in complex product images and the cloud-edge computing framework can provide fast computation speed and evolving algorithm models. A real industrial case study is presented to illustrate the effectiveness of the proposed method. The results show that the proposed method can provide high detection accuracy within a short time.

Keywords: Automated surface inspection, smart product-service system, convolutional neural networks, cloud-edge computing

*Corresponding author: wang yuanbin@zju.edu.cn; junzou@zju.edu.cn

October 22, 2019

1. Introduction

Product quality always plays a critical role for manufacturing companies [1]. Among all different aspects of part quality, surface quality is a key feature that commonly examined by manufacturers. Many manufacturers still rely on intensive workforce to detect the surface flaws of their parts [2]. However, there are several drawbacks on manual inspection. Firstly, the inspection speed is slow and will limit the speed of production line [3]. Secondly, the scalability is low and it requires a long time to train a qualified inspector [1]. Meanwhile, it is not uncommon that humans' performance could be instable after a long working time [1, 3, 4]. Allowing unqualified parts being distributed in the market may cause extra cost to the companies [4] such as logistic fees and negative impact to their brands.

To overcome these problems, automated surface inspection (ASI) has become a hot topic in recent years. Traditional machine vision methods have been widely used for ASI. Common techniques include edge detection [5], histogram-based features [6], thresholding [7], etc. There are various features can be extracted from an image and they need to be carefully crafted for specific tasks [1]. This is time-consuming and can only be used in specific cases [8]. The accuracy may not be enough in more complex situations [1].

With the rapid development of artificial intelligence, deep learning, especially convolutional neural networks (CNNs) have been proposed for ASI problems and shown promising performances [1, 9, 10, 11, 12]. They can automatically extract important features and proceed classification in the same networks. The results could be close to or even surpass human performance [13]. However, these methods usually take an image as the input and directly generate a class that it should belong to. Although a large image can be divided into small pieces for classification, it is difficult to decide the size of the small pieces, especially for geometrically complex products. If the size is too small, a small portion of background mixed in the image may be recognized as a defect due to the lack of surrounding information. If the size is too large, it may involve multiple types of defects and be difficult to locate each of them in the image. Therefore, it is more suitable in the situation that the part surface occupies the most of the image (e.g. flat surfaces). It may be less efficient

to accurately locate the defects when the part becomes complex and the image involves a large portion of the background. The object detection method, on the contrary, combines the item localization and classification in the same networks [14]. It can firstly identify the areas that probably contains an object and then classify them. In this way, the classification can quickly and accurately find the area with defects. This kind of methods is more suitable for the defect detection of complex products.

Although CNN-based methods have been tested in various defect detection scenarios, little study has discussed about how to implement this kind of methods in production environment and create smart services [15]. To ensure the algorithms to perform efficient defect detection and keep evolving with more data accumulated, it is important to combine the hardware (e.g. camera, light source) and software services (e.g. scalable computational resources and updating CNN models) to create more sophisticated ASI systems. Recently, the smart product-service system (SPSS) has been proposed to integrate smart products and e-services to satisfy consumers' various and fast-changing needs [16, 17]. Integrating advanced information technologies including Internet of things and cloud-edge computing, a smarter ASI system could be created.

In this paper, a smart surface inspection system (SSIS) is proposed to take advantage of advanced information and communication technologies (ICT). The system integrates the consumer (i.e. the manufacturer of parts to be inspected), the product provider (SSIS hardware developer) and the service provider (i.e. algorithm developer) in a cloud-edge computing environment to achieve high speed computation performance and continuously improved algorithms. One of the most accurate object detection algorithms, faster R-CNN, is utilized to identify possible defects in part images.

This paper is organized as follows. Section 2 introduces the theoretical background of the proposed system. The system framework is proposed in Section 3. Section 4 explains the faster R-CNN algorithm for defect detection. Section 5 displays a case study using real industrial data. Section 6 concludes the paper.

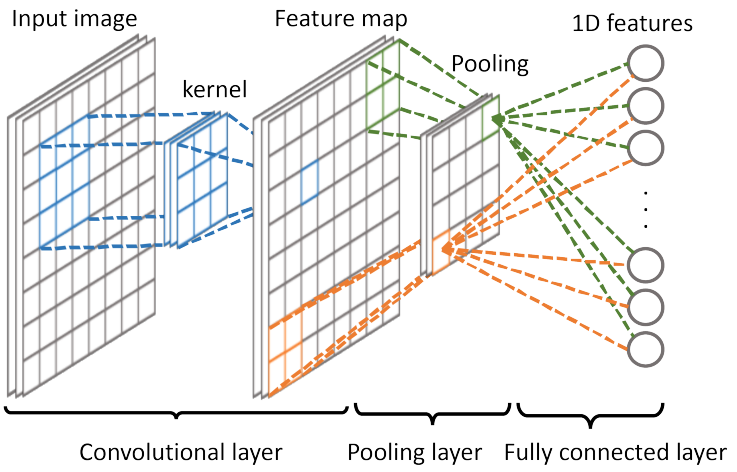

Figure 1: Common types of layers in CNNs

2. Theoretical Background

There are two main aspects of the proposed SSIS: identifying defects in a given image (algorithms) and design services for the system to create more value (servitization). The theoretical background of the proposed system in terms of both aspects are introduced as follows.

2.1. CNN-based Object Detection

The convolutional neural network (CNN) is a typical type of the artificial neural networks (ANNs) [2]. The neurons in CNNs are connected locally instead of globally as in common ANNs. This approach enables the local correlations of neighboring pixels to be considered, the invariance to the location of objects and faster computation speed [18]. Therefore, it can provide better performance in image processing. There are several common types of layers in CNNs as shown in Figure 1 [14].

- **Convolutional layer:** Using a small kernel to convolve the whole image and intermediate feature maps. This step is mainly to allow the kernel to learn features in the image.
- **Pooling layer:** Usually following one or several convolutional layers to reduce the dimension of the feature map and the risk of overfitting.

- **Fully connected layer:** Usually following one or several convolutional and pooling layers to convert 2D feature maps to 1D vectors for classification or other processing.

The CNNs are initially designed for image classification purposes, but it can be adjusted for object detection applications [19]. There are two types of methods: two-stage method and one-stage method. Two-stage methods, including fast R-CNN [20] and faster R-CNN [21], firstly propose a large pool of candidate boxes in an image and then use CNN to determine whether an object is in it or not. This kind of methods usually takes a longer time but yields better results. One-stage methods, including YOLOV3 [22] and SSD [23], only evaluate a small set of predefined boxes in different feature maps with various scales. In this way, it can achieve faster detection speed but this may influence its accuracy. Considering the detection accuracy is more important than speed in ASI, two-stage methods are more suitable in this kind of scenarios. The faster R-CNN is one of the most accurate and fast algorithms in two-stage methods [21]. Therefore, it is applied in this paper for defect detection.

2.2. Smart Product-Service System

Product-service systems combine products and services as single solutions to better serve various consumers [24]. With the rapid development of ICT, products are becoming increasingly smart. The smartness can be presented with the capabilities of autonomy, adaptability, reactivity, multi-functionality, etc. [25]. Possessing one or more of these capabilities, smart PSS can be created to increase the competitiveness of the whole solutions [16, 26]. Various products such as smart wearables and electrical appliances has shown its potential to fulfill consumers' various demands [16, 17]. There are three main elements for a SPSS, i.e. smart & connected products (SCP), smart services (SS), and the smart & connected environment (SCE) they are implemented in (Figure 2) [27, 28]. To avoid misunderstanding, the term "product" in this paper represents the surface inspection equipment. An object to be inspected is called a "part".

The advanced ICT provides possibilities to design more innovative SPSS. On one hand, the emergence of CNN dramatically improved the smartness of image processing systems

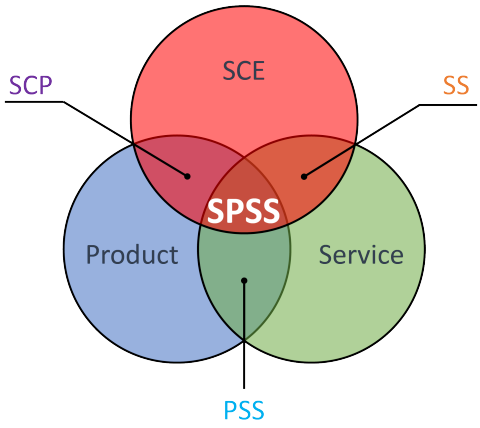

Figure 2: Major elements in SPSS

by automating the feature extraction process [1, 8]. This enables smarter and more sophisticated services to be generated for consumers. On the other hand, the computational frameworks have been increasingly powerful for various requirements and demands. Cloud computing has been dramatically expanded during the past decades due to its vast computational resources and everything as a service paradigm [29]. However, it has its drawbacks on real-time computation and risk of data leaking [30]. To tackle this issue, edge computing has been proposed to proceed some of the computational tasks at the edge of the network close to the data sources to provide faster responses [31, 32]. In this paper, an SSIS in the cloud-edge computing environment is proposed as a new surface inspection system. It connects the consumer, product provider and service provider to generate a flexible and collaborative system that can keep evolving over time.

3. SSIS Framework

There are four main elements in the proposed SSIS framework, including the consumers' production lines as the data sources of SSIS and the three main elements of SPSS (i.e. SCP, SS and SCE as mentioned in Section 2.2). The framework of the proposed SSIS is depicted as in Figure 3.

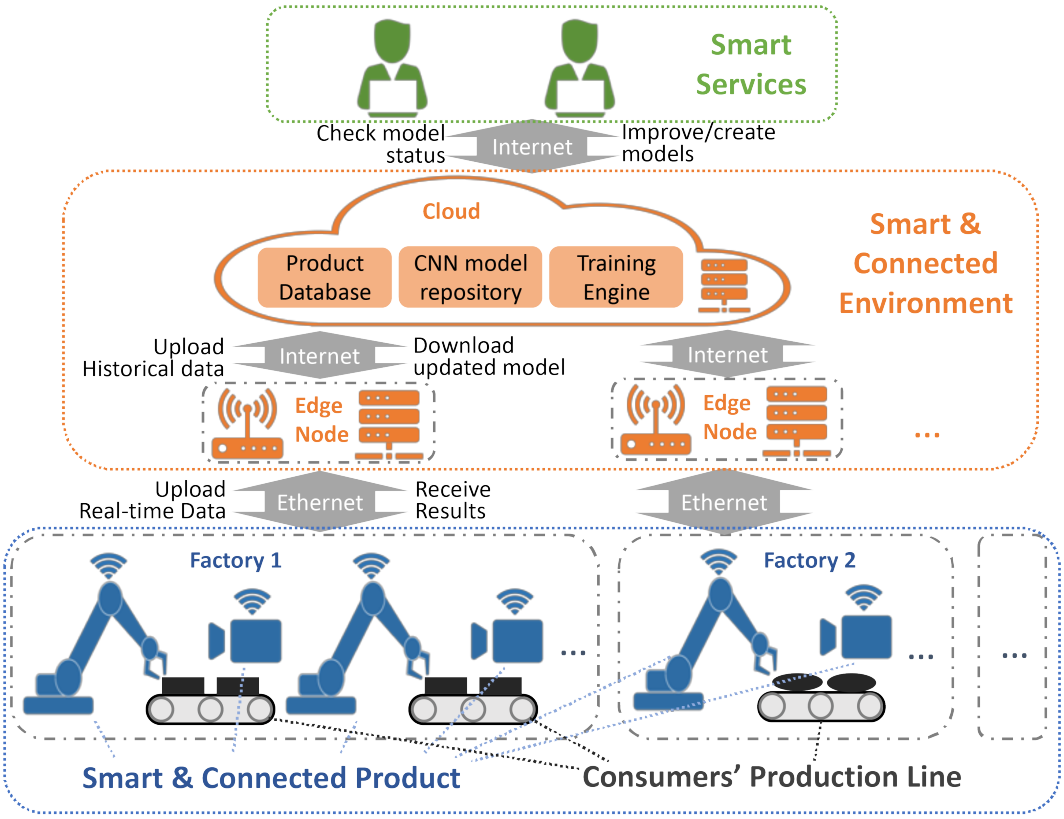

Figure 3: SSIS Framework

3.1. Data sources

For typical ASI systems, the input data are the images of the parts to be examined. The parts usually come from the consumers' production lines to different inspection spots. Different defect detection models may be required in different spots, such as for different parts, for a part at different processing stages, or for a different type of defects. Therefore, it is important for manufacturers to have a central server to store and maintain defect detection models and distribute them to the right inspection spots.

3.2. Smart & Connected Products

In this system, SCP mainly include the hardware to control part's positions, light sources and image sensors to capture image data. The position control system can be used to adjust the position of image sensors instead of parts if necessary. Depending on the complexity of the part, the position control system could be as simple as a sensor to stop the part when

it arrives, or as complex as a robotic arm to adjust a part's location and orientation. The light sources are used to ensure the brightness of images is enough to identify defects. The image sensors (e.g. CCD and CMOS) are used to capture the surface image of parts for defect detection. It could be a single camera to take multiple images of a part or several cameras to take less images depending on specific requirements. CNN-based object detection algorithms require a large computational power, which is difficult to be integrated to an embedded system. Therefore, this equipment, especially the image sensor, are connected to the smart gateways (acting as edge nodes) so that edge nodes can control the actions of these.

3.3. Smart Services

There are two types of services in the proposed system: the cloud-edge computing platform as a service and the defect detection software as a service.

The computing platform service provides sufficient cloud computing and edge computing resources and maintain an efficient communication and collaboration between the cloud and edge ends. In this way, data at the edge nodes can upload to the cloud server when the communication is not busy and the cloud server can implement updated CNN models on related edge nodes. In this way, the inspection results can be generated locally close to the data sources with low latency, saving the time and energy of transmitting data to the remote cloud server. Unlike the embedded system, edge nodes have the capability to scale up the computational power when necessary, which can provide the fastest response from CNN models.

The software service provides the capability to create new or upgrade current CNN models with new data generated from the data sources. When the precision of a CNN model goes down, the system will send an alert to the service provider through the cloud platform. The service providers can then use the new generated data to improve the current model and implement it to the edge nodes through the cloud platform. In this way, the performance of the SSIS can be maintained and keep evolving to deal with new problems. When a new product is involved, the service providers can use its image data to create a

new CNN model for it.

3.4. Smart & Connected Environment

The SCE plays a crucial role to enable all the functions to be smoothly operated as expected. The SCP are connected to the edge nodes via the Ethernet due to its stability and fast speed. In the future, 5G with its capabilities of massive connectivity and very fast speed [33] can be utilized to replace Ethernet when the technology becomes mature. The edge nodes are at the edge of the Internet close to the data sources. It could be a smart gateway for a factory in the Ethernet environment or a base station in the 5G environment. Edge nodes will be equipped with scalable computational resources to deal with real-time requests (i.e. defect detection in this case). In this way, edge nodes can reduce the data processing time and energy (avoid data transmission to the cloud) and be more secured with controllable data exposure to the Internet [31, 29]. The cloud platform provides massive computational power for computation-intensive tasks (i.e. CNN model training in this case). Meanwhile, it can be connected remotely via the Internet so it can act as a central server to manage the algorithms and functions for all connected edge nodes. There is a product database to carry the information of various products and a CNN model repository to carry algorithm models for different products and different types of defects. The algorithm created or revised by services providers will be firstly uploaded to the cloud platform, then distributed to the corresponding edge node.

4. Defect Detection Method Based on Faster-RCNN

In the proposed system, accuracy is one of the most important features since the calculation speed can be improved by increasing the computational power at edge nodes. In practice, the defects could be very small and mixed with other non-defect objects (e.g. background, dust, ink). The geometrical complexity increases the difficulty of accurately identifying defects. Evolved from R-CNN and fast R-CNN, faster R-CNN [21] is one of the most accurate deep learning methods for object detection. In this paper, it is applied to detect defects from product images (Figure 4). There are four main parts: convolutional layer,

region proposal network (RPN) layer, region of interests (ROI) pooling layer, classification layer.

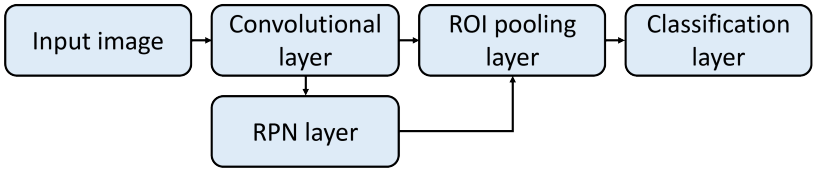

Figure 4: Faster R-CNN computation procedure

4.1. Image Pre-processing

The cameras used for defect detection usually have high resolutions. Therefore, the size of part image would be very large. Although faster R-CNN allows images with any sizes as input, the very large size will influence the computational efficiency of the network. Directly reducing the resolution of the part will lose information in the images. Considering that the defects could be very small areas, reducing resolution may not be suitable in this case. Therefore, a sliding window of 300 $\times$ 300 pixels is used to crop an image into several pieces. Figure 5 shows some examples of sliding windows. There is an overlap between two consecutive windows to ensure the details are intact around the edge area.

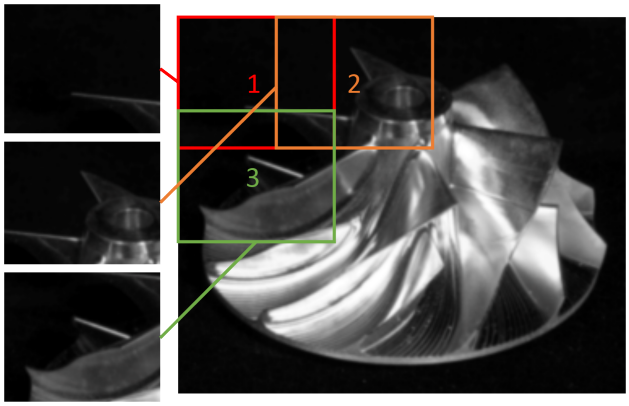

Figure 5: Sliding windows to crop an image

4.2. Defect Detection

The process of defect detection using faster R-CNN is described in Algorithm 1. As one of the most accurate convolutional neural networks, ResNet101 [34] is applied as the

Algorithm 1 Faster R-CNN

Input:

A three-channel images A;

A set of well-trained parameters $\{P_i\}$, where $i \in [1, I]$, I is total number of parameters;

Output:

A set of coordinates $\{x_j, y_j, w_j, h_j\}$ representing the centers and sizes of objects' bounding boxes $\{B_j\}$ in A;

A set of scores $\{S_j\}$ for $\{B_j\}$;

1: Resize A to a predefined size $M \times N$ as B;

$\Rightarrow$ *Convolutional Layer:*

2: Use ResNet101 to extract a set of feature maps $\{C_k\}$ from B, where $k \in [1, K]$, K is the total number of feature maps;

$\Rightarrow$ *RPN Layer*

3: For each C_k, propose $N = 9 \times wf_k \times hf_k$ anchors $\{PA_n\}$, where $n \in [1, N]$, wf_k and hf_k represent the width and height of C_k;

4: Using a Softmax layer to calculate two scores PA_n^obj and PA_n^nobj to represent its possibility of containing or not containing any objects;

5: Adjust the $\{PA_n^x, PA_n^y, PA_n^w, PA_n^h\}$ of the $\{PA_n\}$ using the following equstions:

$$PB_n^x = PA_n^w \cdot d^x(PA_n) + PA_n^x;$$
$$PB_n^y = PA_n^h \cdot d^y(PA_n) + PA_n^y;$$
$$PB_n^w = PA_n^w \cdot exp(d^w(PA_n));$$
$$PB_n^h = PA_n^h \cdot exp(d^h(PA_n));$$

where $\{PB_n\}$ are the predicted boxes after adjusting $\{PA_n\}$, $d^x(PA_n)$, $d^y(PA_n)$, $d^w(PA_n)$, $d^h(PA_n)$ are functions of PA_n, their parameters are pre-trained in $\{P_i\}$;

$\Rightarrow$ *ROI Pooling Layer*

6: Combine the results of $\{PA_n^{obj}, PA_n^{nobj}\}$ and $\{PB_n\}$ to generate the proposal of regions of interests (ROI) $\{R_m\}$, where $m \in [1, M], M \leq N$;

$\Rightarrow$ *Classification Layer*

7: Refine the bounding boxes of $\{R_m\}$ and classify the objects in them using a Softmax layer as outputs.

convolutional layer in the proposed system. A comparison study of the proposed method and other popular object detection methods is conducted in the case study in Section 5.

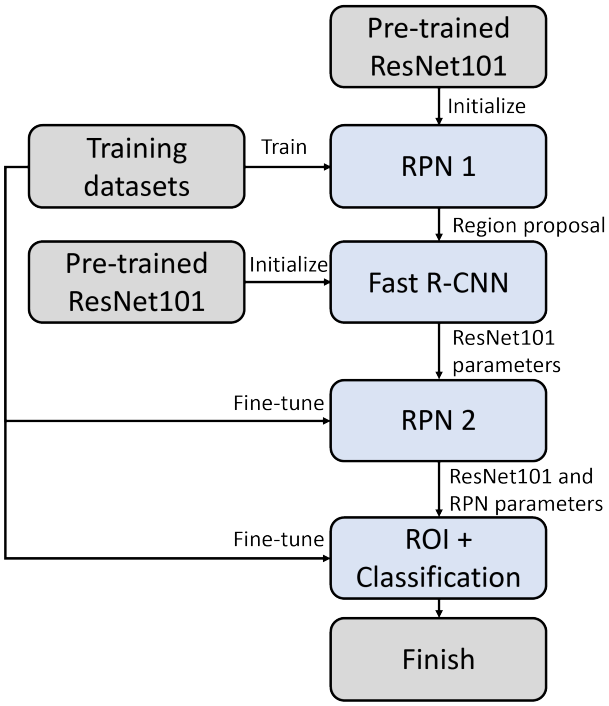

Figure 6: Training faster R-CNN

4.3. Training Process

The training process is similar to traditional neural networks using backpropagation method. More specifically, there are four steps [21] as shown in Figure 6. To accelerate the training process, the convolutional layers can be pre-trained on large common object detection datasets as a start point. For a set of training data, the ground truth boxes of objects in these images and their ground truth classes need to be labeled in advance as training datasets. Meanwhile, a ResNet101 pre-trained by a large general image dataset can be used to initialize the RPN layer and improve the training speed. There are two parts to be trained - RPN layer and fast R-CNN layers (convolutional + ROI + classification).

Regarding to RPN training, to optimize the parameters in both the ResNet101 layers (Step 2 in Algorithm 1) and in the anchor adjustment (Step 5 in Algorithm 1) at the same time, the loss function is designed as [21]:

$$L(\{PA_n^{obj}\},\{t_n\}) = \frac{1}{N_{cls}}\sum_n L_{cls}(PA_n^{obj}, PA_n^*) + \lambda\frac{1}{N_{reg}}\sum_n PA_n^* L_{reg}(t_n, t_n^*) \tag{1}$$

where n is the index of an anchor and PA_n^{obj} is the possibility of the anchor contains an object; $t_n = \{x_n, y_n, w_n, h_n\}$ describes the anchor; The ground truth label PA_n^* is 1 if the anchor contains an object, otherwise 0; t_n^* is the ground truth box; Both the loss functions for classification L_{cls} and for anchor box regression L_{reg} is normalized by their total numbers N_{cls} and N_{reg}.

4.4. Performance Analysis

There are several important index to analyze the performance of an object detection algorithm. The intersection-over-union (IOU) is one of the basic index which can be calculated as

$$IOU = \frac{B_{predict} \cap B_{truth}}{B_{predict} \cup B_{truth}} \tag{2}$$

where $B_{predict}$ and B_{truth} are the predicted bounding box and ground truth bounding box. The higher the IOU value is, the predicted box is more accurate.

Assuming A is the set of predicted boxes and B is the set of ground truth boxes. The possible results are shown in Figure 7. If the predicted object in A is not in the ground truth set B, then it is marked as false positive (FP). If it is in B, then it is marked as true positive (TP). If an object in B is not predicted in A, then it is marked as false negative (FP). According to these values, the precision (P) and recall (R) of an algorithm can be defined as

$$P = \frac{N_{TP}}{N_{FP} + N_{TP}} \tag{3}$$

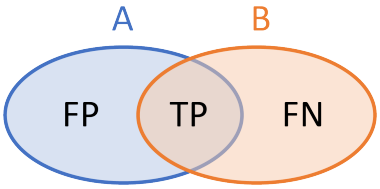

Figure 7: Possible predicted results

$$R = \frac{N_{TP}}{N_{FN} + N_{TP}} \tag{4}$$

where N_{TP}, N_{FP}, N_{FN} are the total number of TP, FP, FN.

5. Industrial Case Study

Turbo blade (as shown in Figure 8) is one of the key components in the turbo engines of the automobile. It works under very high speed of rotation so any small surface defects may influence its dynamic balancing in the working conditions. Furthermore, the geometry of turbo blades is very complex, which makes the defect detection more challenging. Therefore, it is chosen in this paper to illustrate the effectiveness of the proposed method. In this paper, some real data from a turbo blade manufacturer is gathered and tested. This test mainly focuses on the scratch defect as it one of the most commonly existed ones.

There are 81 images of turbo blades with 156 defects collected in total. In this paper, 64 images with 120 defects are used for training and the remaining 17 images with 36 defects are used for verification.

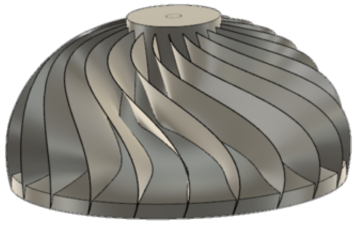

Figure 8: An example of turbo blade model in car engines

5.1. System Implementation

The system structure is displayed in Figure 9. The part images are generated at Hangzhou, China. As the size of images is very large, they are firstly divided and resized to 300×300 pixels smaller images. The Xuelang cloud computing infrastructure is located at Wuxi, China and it is used for model training. The object detection models are pre-trained using MS COCO datasets.

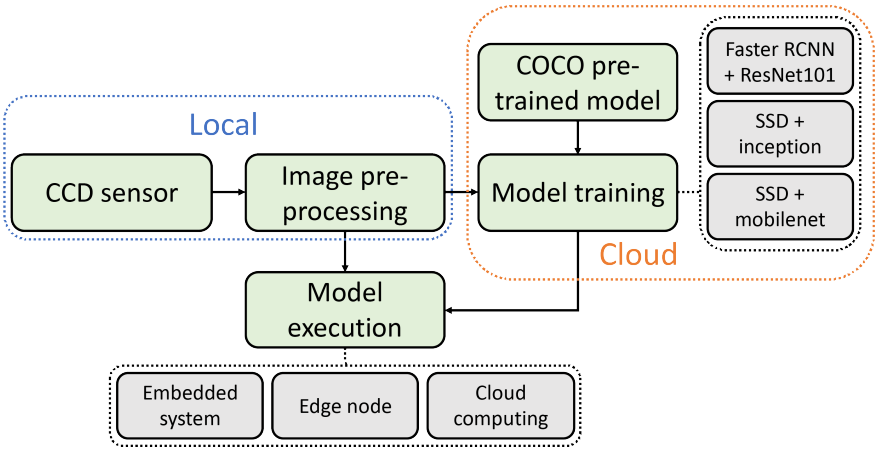

Figure 9: System implementation in the turbo blade case

The setup of the hardware is presented in Figure 10. A robot was used to pick up the turbo blade and rotate it for image capturing. A CCD image sensor with 12 million pixels is used for part image capturing. The camera is connected to the edge node through a network cable. The edge node equipped with an Intel i7-4790 at 3.6Hz processor and a Nvidia GeForce GTX TITAN X GPU. The software was running in the Ubuntu 16.04 OS and the algorithms were developed in the Python 3.6 environment based on Tensorflow.

5.2. Algorithm Comparative Study

Two experiments are conducted to illustrate the performance of the proposed method in terms of detection accuracy and computational speed respectively.

5.2.1. Detection Accuracy

Due to the complexity of the turbo blade geometry and defect appearance, it is not suitable to use traditional machine vision methods for defect detection. Therefore, the

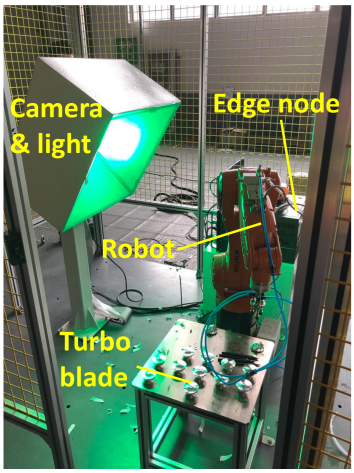

Figure 10: Hardware setup of the test system

proposed faster R-CNN method is compared with another widely used object detection method - Single Shot MultiBox Detector (SSD) [23]. The SSD can be used with different types of convolutional layers. Among them, the inception v3 [35] and mobilenet [36] are the most commonly used ones. Therefore, the detection accuracy of the proposed faster R-CNN is compared with SSD + inception v3 [35] and SSD + mobilenet [36] (Figure 9). The mean average precision (mAP@50) in the training process is displayed in Figure 11. The mAP@50 is a common object detection measurement considering the precision and recall when IOU is larger than 50%. As shown in Figure 11, SSD networks converge fast at the beginning but they stop improving at less than 60% for SSD inception and 30% for SSD mobilenet respectively. Faster R-CNN converges not very fast at beginning but it achieves around 70% at last.

After training process, these models are tested on the verification dataset. The performance of these models is listed in Table 1. The SSD with inception can provide a high correctness of defects identified (93%) but it ignores a lot of actual defects with only 36% found. This is not acceptable to let too many products with defects pass the test. The SSD with mobilenet, in the contrary, found as many defects as faster R-CNN but it identified a lot of defects that should not be (33% precision). This brings extra works for inspectors, which impact its efficiency to reduce workforce. The faster R-CNN found 72% defects with 81% of the objects identified are correct. Some defects identified are displayed in Figure 12.

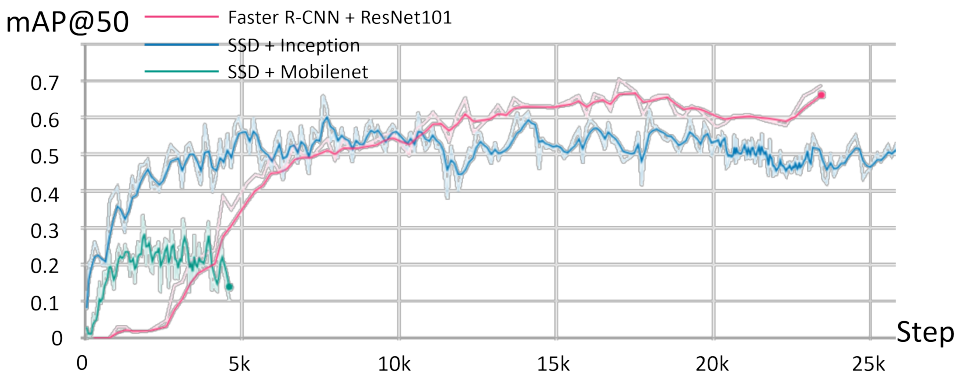

Figure 11: mAP@50 trends during training process

Table 1: Comparison of prediction results

	True positive (TP)	**False negative (FN)**	**False positive (FP)**	**Precision (P)**	**Recall (R)**
Faster R-CNN + ResNet101	26	10	6	0.81	0.72
SSD + inception	13	23	1	0.93	0.36
SSD + mobilenet	26	10	53	0.33	0.72

This is a good balance between the capability of finding defects and the correctness of the defects identified.

The highest precision that can be achieved is influenced by the nature of defects. They are usually very small (Figure 13(a)) and sometimes very light (Figure 13(b)), which makes it difficult to differentiate them from dirts and other objects. Therefore, it is more difficult to achieve high recalls as large object detections. This also causes the algorithm to treat dirt as defects sometimes as shown in Figure 14, which influence the precision. Besides, it is very difficult to get sufficient data to train a better model as defects rarely occur. What is more, the definition of defects is arbitrary to some extent. It may be a defect for one inspector but not be one for another inspector. This may influence the labeling process and further impact the trained model. Despite these challenges, this result has shown the potential to use faster R-CNN method as an assistant tool to help inspectors improve the efficiency of complex part defect detection. The performance can be improved in the detection process with more data accumulated.

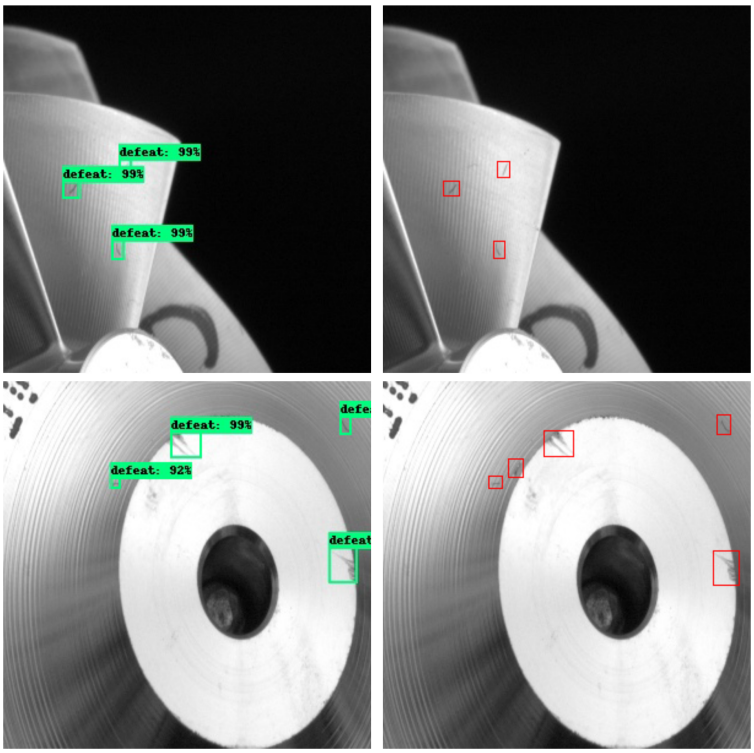

Figure 12: Some results of faster R-CNN (predicted boxes are green and ground truth are red)

5.2.2. Computation Speed

The computation speed of the proposed cloud-edge computing framework is compared with an embedded system and the cloud computing (Figure 9). In the detection process, the computation is conducted at the edge node. In this test, a Nvidia GeForce GTX TITAN X with 11GB memory is deployed in the edge computing device for fast parallel computation. A Raspberry Pi with 1GB RAM CPU is used as an embedded system. The Xuelang cloud platform is used as the cloud computing resources. The detection speed is tested for 100 images and the average computation time per image is presented in Figure 15. It shows that the edge computing method can dramatically increase the computation speed comparing to embedded system (more than 10 times). Although the cloud platform may provider stronger computation power, the image transmission costs extra time and energy and it is influenced by the Internet connection status. Considering each part may need to take more than 10 images to cover all important surfaces and there may be multiple inspection spots, the edge computing method is able to provide fast and stable computation.

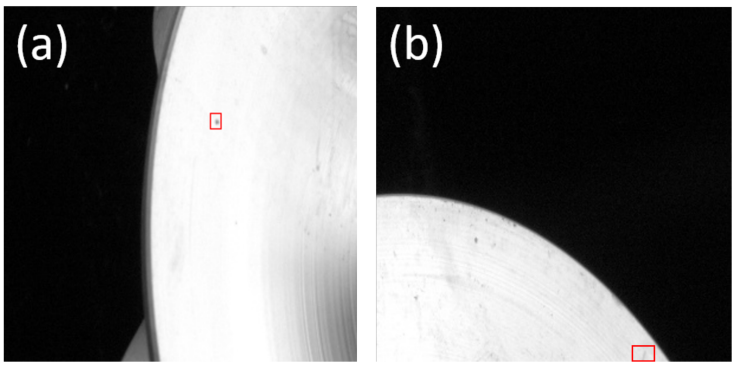

Figure 13: Unidentified defects (a) too small (b) too light

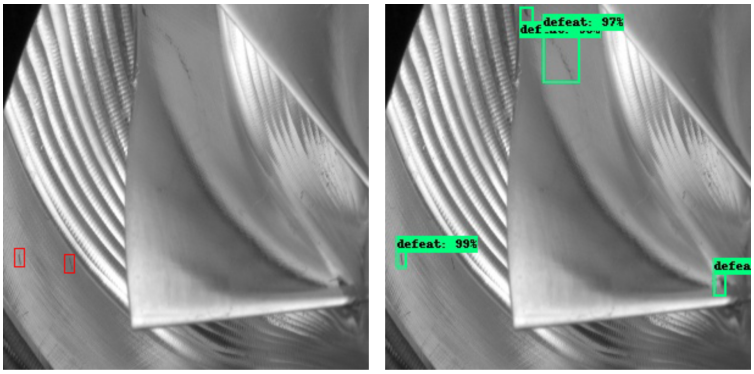

Figure 14: Misidentified non-defect as defect (predicted in green and ground truth in red)

Comparing three networks, faster R-CNN takes more time than other two SSD-based networks. This is caused by different convolutional layers and region proposal methods. The differences can be dramatically reduced by using edge computing method.

6. Conclusion

Automated surface inspection is very important for manufacturing companies to improve the reliability and efficiency of product surface inspection. Embracing advanced information and communication technologies, a smart surface inspection system (SSIS) is proposed in this paper to realize intelligent and fast surface inspection and enable the detection services to be improving with more data accumulated. Key contributions are as follows:

- A cloud-edge computing enabled SSIS is proposed as a new ASI framework to facilitate its implementation in practice. It provides a smart and connected environment to allow

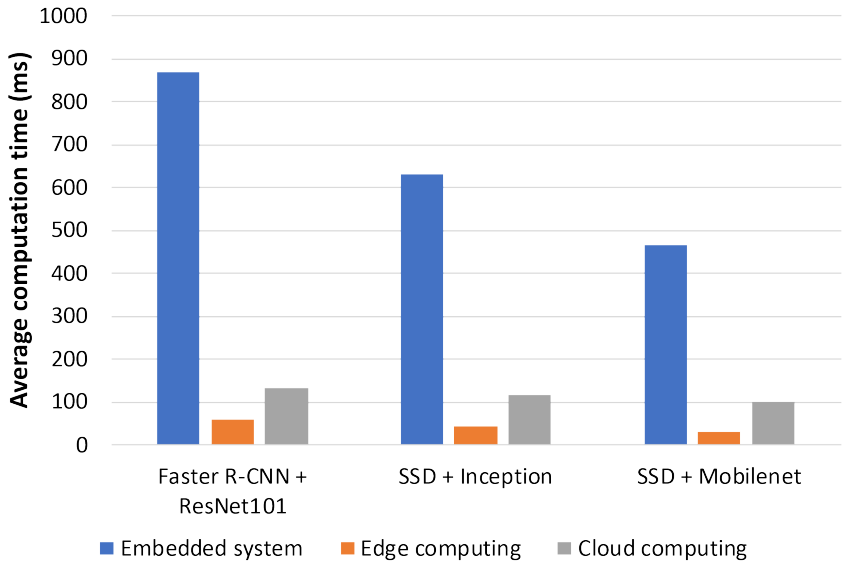

Figure 15: Comparison of computation time

fast detection speed and continuous improvement of detection services.

- Several popular object detection networks have been tested on an industrial dataset. The result shows that the faster R-CNN with ResNet101 can provide the highest recall with high precision for defect detection of complex products.

Although the proposed system has shown a promising performance, there are still several aspects to be improved in the future. Firstly, this paper mainly tested the proposed method on the turbo blade scratch defect. More data for different types of defects need to be gathered to test the method on other aspects. Secondly, defects are often small objects. Optimizing the network for small object detection may further improve the detection performance. Thirdly, the defect datasets are usually difficult to obtain from manufacturing companies due to its low possibility of occurrence. How to train a robust and more accurate detection model with limited data is still a challenging task.

Fundings

This work was supported by the National Natural Science Foundation of China [Grant No. 51890885].

References

[1] D. Weimer, B. Scholz-Reiter, M. Shpitalni, Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection, CIRP Annals - Manufacturing Technology 65 (1) (2016) 417–420. doi:10.1016/j.cirp.2016.04.072.

[2] T. Wang, Y. Chen, M. Qiao, H. Snoussi, A fast and robust convolutional neural network-based defect detection model in product quality control, International Journal of Advanced Manufacturing Technology 94 (9-12) (2018) 3465–3471. doi:10.1007/s00170-017-0882-0.

[3] K. D. Joshi, V. Chauhan, B. Surgenor, A flexible machine vision system for small part inspection based on a hybrid SVM/ANN approach, Journal of Intelligent Manufacturing (2018). doi:10.1007/s10845-018-1438-3.

[4] E. Weigl, W. Heidl, E. Lughofer, T. Radauer, C. Eitzinger, On improving performance of surface inspection systems by online active learning and flexible classifier updates, Machine Vision and Applications 27 (1) (2016) 103–127. doi:10.1007/s00138-015-0731-9.

[5] H. Kong, J. Yang, Z. Chen, Accurate and Efficient Inspection of Speckle and Scratch Defects on Surfaces of Planar Products, IEEE Transactions on Industrial Informatics 13 (4) (2017) 1855–1865. doi:10.1109/TII.2017.2668438.

[6] Ç. Aytekin, Y. Rezaeitabar, S. Dogru, I. Ulusoy, Railway fastener inspection by real-time machine vision, IEEE Transactions on Systems, Man, and Cybernetics: Systems 45 (7) (2015) 1101–1107. doi:10.1109/TSMC.2014.2388435.

[7] M. Win, A. R. Bushroa, M. A. Hassan, N. M. Hilman, A. Ide-Ektessabi, A contrast adjustment thresholding method for surface defect detection based on mesoscopy, IEEE Transactions on Industrial Informatics 11 (3) (2015) 642–649. doi:10.1109/TII.2015.2417676.

[8] J. Wang, Y. Ma, L. Zhang, R. X. Gao, D. Wu, Deep learning for smart manufacturing: Methods and applications, Journal of Manufacturing Systems 48 (2018) 144–156. doi:10.1016/j.jmsy.2018.01.003.

[9] J. K. Park, B. K. Kwon, J. H. Park, D. J. Kang, Machine learning-based imaging system for surface defect inspection, International Journal of Precision Engineering and Manufacturing - Green Technology 3 (3) (2016) 303–310. doi:10.1007/s40684-016-0039-x.

[10] S. Mei, H. Yang, Z. Yin, An unsupervised-learning-based approach for automated defect inspection on textured surfaces, IEEE Transactions on Instrumentation and Measurement 67 (6) (2018) 1266–1277. doi:10.1109/TIM.2018.2795178.

[11] R. Ren, T. Hung, K. C. Tan, A Generic Deep-Learning-Based Approach for Automated Surface Inspection, IEEE Transactions on Cybernetics 48 (3) (2018) 929–940. doi:10.1109/TCYB.2017.2668395.

[12] Y. Wang, Y. Lin, R. Y. Zhong, X. Xu, IoT-enabled cloud-based additive manufacturing platform to support rapid product development, International Journal of Production Research (2018) 1–

17doi:10.1080/00207543.2018.1516905.

[13] Z. A. Siddiqui, U. Park, S. W. Lee, N. J. Jung, M. Choi, C. Lim, J. H. Seo, Robust powerline equipment inspection system based on a convolutional neural network, Sensors 18 (11) (2018) 1–25. doi:10.3390/s18113837.

[14] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, M. S. Lew, Deep learning for visual understanding: A review, Neurocomputing 187 (2016) 27–48. doi:10.1016/j.neucom.2015.09.116.

[15] Y. Wang, K. Hong, J. Zou, T. Peng, H. Yang, A CNN-based visual sorting system with cloud-edge computing for flexible manufacturing systems, IEEE Transactions on Industrial Informatics (2019) 1–10doi:10.1109/TII.2019.2947539.

[16] P. Zheng, T. J. Lin, C. H. Chen, X. Xu, A systematic design approach for service innovation of smart product-service systems, Journal of Cleaner Production 201 (2018) 657–667. doi:10.1016/j.jclepro.2018.08.101.

[17] A. Valencia, R. Mugge, J. P. L. Schoormans, R. Schifferstein, The Design of Smart Product-Service Systems (PSSs): An Exploration of Design Characteristics, International Journal of Design 9 (1) (2015) 13–28.

[18] M. D. Zeiler, Hierarchical Convolutional Deep Learning in Computer Vision, PhD Thesis, New York University (2013).

[19] A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis, Deep Learning for Computer Vision: A Brief Review, Computational Intelligence and Neuroscience 2018 (2018) 1–13. doi:10.1155/2018/7068349.

[20] R. Girshick, Fast R-CNN, in: International conference on computer vision, 2015, pp. 1440–1448. doi:10.1109/ICCV.2015.169.

[21] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (6) (2017) 1137–1149. doi:10.1109/TPAMI.2016.2577031.

[22] J. Redmon, A. Farhadi, YOLOv3: An Incremental Improvement (2018) 1–6arXiv:1804.02767.
URL http://arxiv.org/abs/1804.02767

[23] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, A. C. Berg, SSD: Single shot multibox detector, in: Computer Vision – ECCV 2016, Vol. 9905, 2016, pp. 21–37. arXiv:1512.02325, doi:10.1007/978-3-319-46448-0_2.

[24] M. J. Goedkoop, C. J. van Halen, H. R. te Riele, P. J. Rommens, Product service systems, Ecological and Economic Basics (1999).
URL https://teclim.ufba.br/jsf/indicadores/holan Product Service Systems main report.pdf

[25] E. J. Hultink, S. Rijsdijk, How Today's Consumers Perceive Tomorrow's Smart Products, Journal of Product Innovation Management 26 (1) (2009) 24–42. doi:10.1111/j.1540-5885.2009.00332.x.

[26] P. Zheng, C. H. Chen, S. Shang, Towards an automatic engineering change management in smart product-service systems – A DSM-based learning approach, Advanced Engineering Informatics 39 (August 2018) (2019) 203–213. doi:10.1016/j.aei.2019.01.002.

[27] X. Jin, S. Yu, P. Zheng, Q. Liu, X. Xu, Cloud-based approach for smart product personalization, Procedia CIRP 72 (2018) 922–927. doi:10.1016/j.procir.2018.03.256.

[28] P. Zheng, Y. Lin, C. H. Chen, X. Xu, Smart, connected open architecture product: an IT-driven co-creation paradigm with lifecycle personalization concerns, International Journal of Production Research 0 (0) (2018) 1–14. doi:10.1080/00207543.2018.1530475.

[29] S. Satpathy, B. Sahoo, A. K. Turuk, Sensing and Actuation as a Service Delivery Model in Cloud Edge centric Internet of Things, Future Generation Computer Systems 86 (2018) 281–296. doi:10.1016/j.future.2018.04.015.

[30] G. S. Aujla, N. Kumar, A. Y. Zomaya, R. Rajan, Optimal Decision Making for Big Data Processing at Edge-Cloud Environment: An SDN Perspective, IEEE Transactions on Industrial Informatics 14 (2) (2017) 1–1. doi:10.1109/TII.2017.2738841.

[31] X. Wang, L. T. Yang, X. Xie, J. Jin, M. Jamal Deen, A Cloud-Edge Computing Framework for Cyber-Physical-Social Services, IEEE Communications Magazine 55 (11) (2017) 80–85. doi:10.1109/MCOM.2017.1700360.

[32] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, C. Liu, Fog Computing for Energy-aware Load Balancing and Scheduling in Smart Factory, IEEE Transactions on Industrial Informatics 14 (10) (2018) 4548–4556. doi:10.1109/TII.2018.2818932.

[33] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, J. C. Zhang, What Will 5G Be?, IEEE Journal on Selected Areas in Communications 32 (6) (2014) 1065–1082. doi:10.1109/JSAC.2014.2328098.

[34] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: computer vision and pattern recognition, 2016, pp. 770–778. doi:10.1109/CVPR.2016.90.

[35] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the Inception Architecture for Computer Vision, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Las Vegas, United States, 2016, pp. 2818–2826. doi:10.1109/CVPR.2016.308.

[36] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv: Computer Vision and Pattern Recognition (2017). arXiv:1704.04861.
URL http://arxiv.org/abs/1704.04861